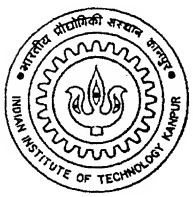
Multi-Objective Optimization of a Biped Locomotion

A Thesis Submitted
in Partial Fulfillment of the Requirements
for the Degree of
Master of Technology

by

Kuldipkumar Patel



to the

Department of Computer Science & Engineering
Indian Institute of Technology, Kanpur

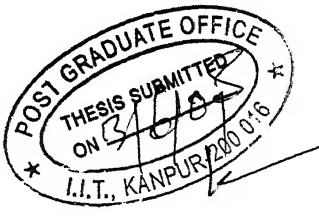
July, 2005

Certificate

This is to certify that the work contained in the thesis entitled " *Multi-Objective Optimization of a Biped Locomotion* ", by *Kuldipkumar Patel*, has been carried out under my supervision and that this work has not been submitted elsewhere for a degree.

May, 2005

(Dr. Amitabha Mukherjee)
Department of Computer Science & Engineering,
Indian Institute of Technology,
Kanpur.



Abstract

A biped learning to walk needs to balance the separate objectives of stability versus distance covered. In this work, we report some initial results from a multi-objective optimization approach for a seven-link biped robot with contact sensors in the feet. We use a fully-connected recurrent neural network as a gait generator, which takes as input the joint angles and the sensor output, and generates the joint torques. The weights of the network are optimized using the multi-objective genetic algorithm NSGA-2. We present different controller designs that results in different non-dominated solutions for the robots having different degrees of freedom.

Acknowledgements

I take this immense opportunity to express my sincere gratitude toward my supervisor Dr. Amitabha Mukherjee for his invaluable guidance. It would have never been possible for me to take this thesis to completion without his innovative ideas and his relentless support and encouragement. I consider myself extremely fortunate to have had a chance to work under his supervision. Inspite of his hectic schedule he was always approachable and took his time off to attend to my problems and give the appropriate advice. Learning how to do research and how to write technical reports from him has being a life time experiences in itself. It has been a very enlightening and enjoyable experience to work under him.

I would like to express my sincere gratitude to Dr. Pabitra Mitra, Dr. Ashish Dutta, Mr. Maitray Srivastava and Mr. Madan Mohan for their invaluable help during the project work.

I also wish to thank whole heartily all the faculty members of the Department of Computer Science and Engineering for the invaluable knowledge they have imparted to me and for teaching the principles in most exciting and enjoyable way. I also extend my thanks to the technical staff of the department for maintaining an excellent working facility.

My stay at IITK was unforgettable to say the least, and the biggest reason for it being my classmates of the great mtech2003 batch. A cream batch with varieties, lots of fun, lots of birthday treats, job parties and photo sessions in the campus. I thank my classmates just for being such great buddies.

Contents

List of Figures

Chapter 1

Synthesizing Walking

1.1 Walking

Nature has solved efficiently the problem of biped locomotion during the long evolution of human species. Therefore, the analysis of basic anthropomorphic walking may be a source of information of great interest to tackle the question of biped robot walking control. Bipedal robots can navigate into the areas which are normally inaccessible to wheeled robots. This kind of robots can be used as walking aids for paraplegics or limb amputees, to inspect dangerous environments, in agricultural work and entertainment industry. So research in biped robotics is increasing with a rapid rate[1].

Walking is one of the routine actions performed by humans. Our eyes could instantly discriminate if something does not look quite right when we see someone walking with a limp or when a computer animation is incorrectly modeling some creature. Natural looking human locomotion is an important problem to computer simulation and animation, but it is a difficult problem to solve.

To start with, We shall explain precisely what static and dynamic equilibrium mean for biped robot [2]. When we apply some pushing force to a standing robot, it has to compensate for the displacement induced by the disturbance. To maintain its static equilibrium, it has to balance using the frictional forces of the ground.

When the disturbance becomes too large, the contacts can be insufficient to fight the displacement. The fall is then avoided only if the biped makes a step forward. Static equilibrium can be evaluated with the distance between the CoG[1] projection and the support base contour. For dynamic equilibrium such a criterion does not exist. However, the position of the ZMP[2] equivalent to the center of pressure of contact forces is widely used. Dynamic equilibrium is guaranteed when the ZMP is inside the support polygon.

1.1.1 Static Walking

Robot is statically stable that is the basic assumption in static walking. Statically stable means that if we don't apply any kind of force except gravity then the robot will stay indefinitely in a stable position. For that it is necessary that the projection of the CoG should remain in the support area as shown in the Figure 1.1. The support area is minimum convex polygon in case of both the feet on the ground or the foot surface in case of one foot is in contact with the ground. These are referred to as single and double support phases, respectively. Also, walking speed must be low so that inertial forces are negligible. This kind of walking requires large feet, strong ankle joints and can achieve only slow walking speeds. One such robot was developed by Kulkarni [3].

1.1.2 Dynamic Walking

Dynamic walking meaning maintaining the CoG inside the support polygon over the course of time, hence biped dynamic walking allows the CoG to be outside the support region for limited amount of time. There is no absolute criterion that determines whether the dynamic walking is stable or not. Indeed a walker can be designed to recover from different kinds of instabilities. However, if the robot has ankle joints and always keeps at least one foot flat on the ground then the ZMP can

[1]The center of gravity of an object is a point at or near the object through which the resultant weight of the object passes.

[2]The ZMP(Zero Moment Point) is the point about which sum of all forces and moments of all the masses of the biped is zero.

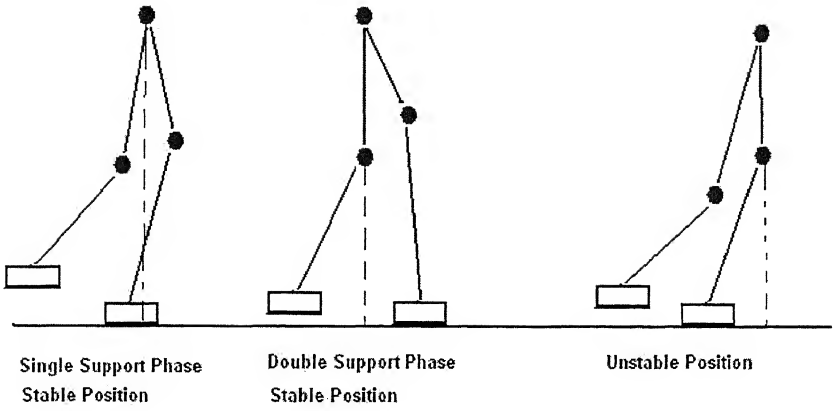


Figure 1.1: Static Walking

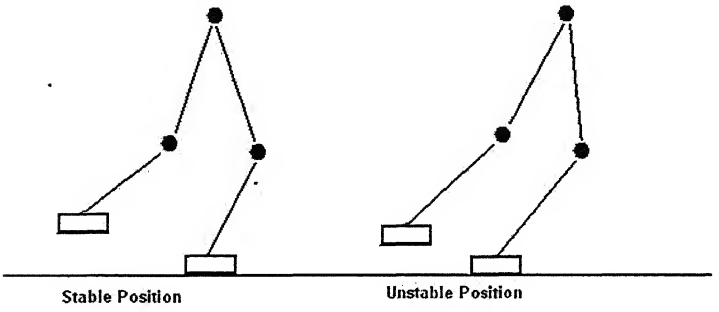


Figure 1.2: Dynamic Walking

be used as a stability criterion. As long as the ZMP is inside the support region the walking is considered dynamically stable because it is the only case where the foot can control the robot's posture. Dynamic walking is achieved by ensuring that the ZMP is always inside the support polygon [3] as shown in figure 1.2.

1.2 Walking in humans

Various invariants have been extracted from walking structure. In steady state, walking is symmetric and periodic. The walking cycle can be divided into strides which are themselves divided into steps. Each cycle is composed of single and double

supports. The walking structure can vary from one individual to another depending on their sex, age, weight and height. A same individual also presents various walking patterns depending on learning and tiredness, which are a source of modification of the gait.

The major requirements for successful locomotion are [4]:

1. Production of locomotor rhythm to ensure support to body against gravity
2. Production of muscle forces that will result with frictional force required for propelling in the desired direction
3. The moving body must be dynamic equilibrium
4. The movements must be able to adapt to meet environmental demands and the tasks selected by the individual animal.

For an animal to have successful locomotion then all the above four must be satisfied. Humans satisfy the above and their locomotor learning also, described in next section.

Humans are the most successful bipeds on Earth. Hence the way that they move is of great interest to many disciplines including robotics. The rhythmic behaviors that they exhibit have been proved to be controlled by central pattern generators. Humans learn to walk through maturation and experience. The capacity to walk is genetically available and can be seen even in new born babies. The development of locomotion in humans takes the following steps [4]

1. Child lifts head
2. Child supports its own body on its own arms
3. Child has ability to turn over
4. Child can sit up
5. Child can walk with assistance

6. Child can walk alone

The above steps show that learning to walk is a very complex procedure. Throughout these steps the neural circuits are forming and eventually in step 6 a central pattern generator controller can successfully control the locomotion of a child.

The successive stages involved in the walking control process can be summarized as follows: The central nervous system generates a signal for the displacement; The transmission of this signal to the muscles via spinal cord; muscle movements required in the locomotion; regulation of the forces and joint torques by the central nervous system on the basis of sensor information.

In our work we use such a nervous system called Artificial Neural Networks and it is being evolved using an evolutionary algorithm, the central nervous systems in the child also passes through evolutionary stage till the child learns walking properly. ANN also takes inputs from the foot sensors as the biological nervous system takes sensor information in humans.

1.3 Related Work

As we have discussed in previous section Bipedal locomotion is a great challenge in autonomous robotics. Many problems on equilibrium and control strategies have to be solved to design an efficient walking robot. In order to design humanoid robots with a human-like gait, many researchers tried to investigate how the human nervous system is controlling bipedal locomotion and how one can model it with efficient controllers.

The CPG-based controllers inspired by biology. Central Pattern Generators (CPGs) are small neural networks situated in the spinal cords of vertebrate animals. With simple and low-dimensional control inputs from higher parts of the brain, they are able to generate all the rhythmic control signals needed to coordinate the muscles in complex movement tasks [5]. Hamid Benbrahim had been

successful in applying "Reinforcement Learning" for a CGP driven biped robot [5]. The approach of RL for CGP controller was inspired by the control mechanism of animals. He had used an actor critic model where the actor performs the action and critic gives the reward. The output of this type of control system is the joint angles and not the desired torques which requires inverse kinematics to be solved.

To make the robot walking smoothly one will have to solve the whole set of dynamic equations which is very tedious job and also asks for redoing the whole work with the change in the morphology of the robot so the concept of evolutionary computation was introduced to remove some part of solving dynamic equations set. Jonathan Roberts, Damien Kee and Gordon Wyeth[6] had obtained improved joint control using a genetic algorithm for a humanoid robot. The robot having 23 DOF was used in the experiment and GA was used to tune the PI controller for lower 15 joints. They kept *tracking error* and *smoothness* were the fitness criteria. They have got far better results with GA compare to hand coded tuning. Leading researchers in the field of robotics Endo, Kitano et al. have used GA in optimizing biped locomotion of a humanoid called PINO [7]. They had assumed the locomotion of right and left legs are symmetry and periodical with phase difference of π . They adopted combination of sinusoidal and cosine function to represent such locomotion and optimized the parameters of the equation with GA. The fitness function was combination of distance covered and energy consumption. The evolutionary algorithms try to land on an optimal solution in a huge search space. If we map the habitat of the robot to this search space then we can get the optimal solution to avoid the obstacles in the path of a robot using such evolutionary computation approach. Abraham Howell [8] used such evolutionary computation approach called Genetic Programming [9] to evolve a suitable obstacle avoidance program for a low-cost wheeled mobile robot equipped with Blue tooth connectivity.

The introduction of learning and neural networks to biped locomotion has shown better results than conventional control methods. Indeed, it is difficult to accurately model the dynamics of a biped and to find analytical control rules that will solve

stability and nonlinearity problems. At School of Electronic and Electrical Engineering, The Robert Gordon University, M Maxell et al.[10] used an artificial nervous system for animats, a class of robots based on animals. The system is based on a hierarchical model and has been implemented using evolutionary artificial neural networks. A new neuron model has been developed for use in the system. The artificial neural networks [11] are combined in a flexible manner to create central pattern generators which control the gait of the animat during locomotion. Bipedal walking gaits have successfully been created using this model.

The combined approach of neural networks and evolutionary computation gives user a complete relaxation from the whole dynamic analysis work. In this approach the network works as a central pattern generator and the weights of the network are being optimized using evolutionary algorithms. If one can include the morphology of the robot with the weights of the network then this co-evolutionary approach can ease the design part of the robot. Endo, Kitano et al. came up with such a new approach of co-evolution of morphology and the walking pattern of biped robot [12]. They had used neural oscillator as a controller and the controller was evolved using Genetic Algorithms. Previously the robots were designed using trial and error method and the morphology of the robot was being assumed while designing the controller but in this technique morphology and locomotion are considered simultaneously, so we do not need to spare time with trial and error method. At the University of Zurich, Josh Bongard and Chandana Paul have tried co-evolution of morphology and locomotion with sensor feedbacks for a biped having six degrees of freedom [13]. They have used a fully connected recurrent neural network as a controller and evolved it with a real-coded Genetic Algorithms [14]. The network takes the joint angles and the output of the foot sensors as input and throws the torques as output applied to the respective joints. The same group of researchers have decoupled the controller for each of the legs and let the controller evolved using real-coded Genetic Algorithms [15]. The architecture of the neural controller is almost same as the one in previous work. They also have been successful in sensorimotor control of biped locomotion based on contact information [16]. The controller

they have used was purely a sensor based neural network and they obtained positive results for an 8-DOF biped.

1.4 Humanoids across the globe

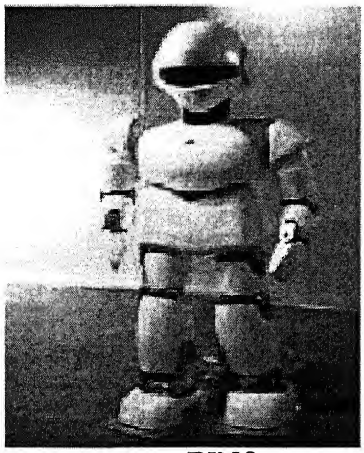
Some Humanoid projects were the motivation of this work discussed in the subsequent paragraphs.

Kitano Symbiotic Systems developed the humanoid called PINO[17]. It was designed to be a platform for research in robotics and AI. There were four major issues in Pino's design

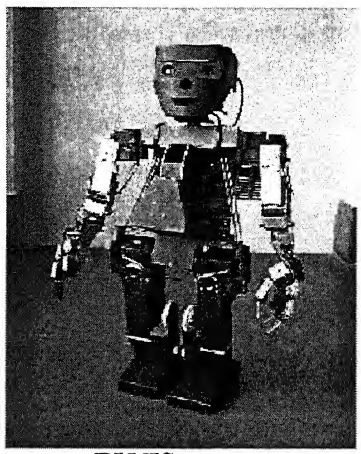
- high DOF system to realize various behaviors
- exterior design
- cheap mechanical components
- A basic behavioral control systems

Each leg of PINO has 6 DOFs, each arm has 5 DOFs, the neck has 2 DOFs and the trunk has 2 DOFs, hence 26 DOFs in total. Pino has 26 motors which correspond to the number of DOFs. And it has various kinds of sensors, For example, visual sensor for recognizing objects, posture sensor for detecting its body's balance, force and tactile sensor for detecting contact to others and falling down, and so on. The controller mechanism is having servo modules. PINO has eight force sensors also.

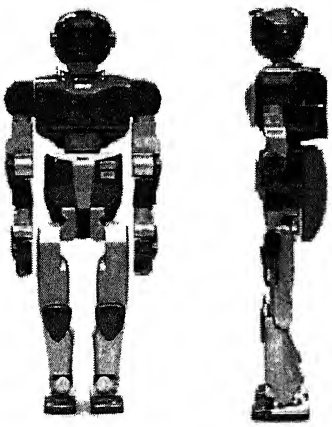
ELVIS humanoid was developed at Chalmers University of Technology, Department of Physical Resource Theory, Complex Systems Group, Sweden [18]. He is a bipedal robot with human-like geometry and motion capabilities. He is also the first robot in a series of planned humanoid experiments, all of which will be primarily controlled by evolutionary adaptive methods. The work started in 1998 and He took his first autonomous steps in April 2000 and participated in Expo2000 in Hannover, Germany, the following summer. The final goal of this project was to build a human-sized robot based on a plastic human skeleton to ensure geometric authenticity. ELVIS is a scaled model with a height of about 60 cm built with 42



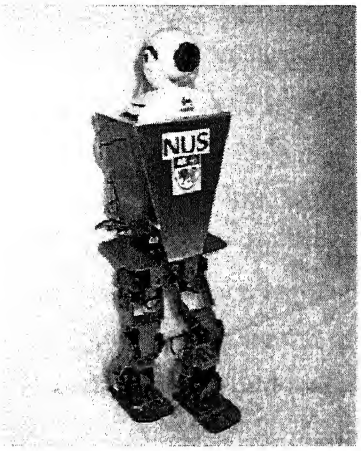
PINO



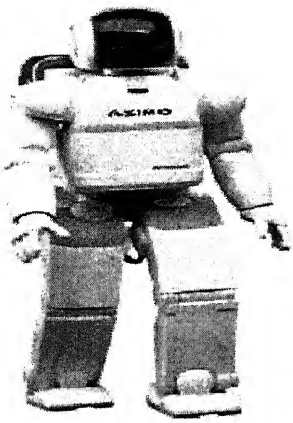
ELVIS



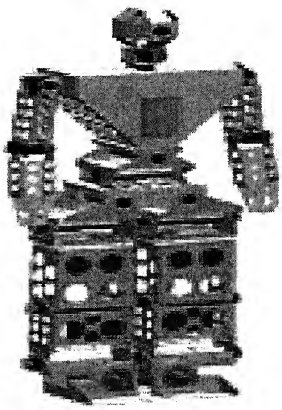
HRP



RoboSapien



ASIMO



GuRoo

Figure 1.3: Elite Humanoid Projects

servos giving a high degree of freedom in legs, arms and hands. The robot is guided by microphones, cameras and touch sensors.

HRP humanoid was developed at Kawada Industries, Inc. Japan in 2003[19]. It is the first humanoid robot that has the same size as a human and that can lie down and get up. The features of the hardware are a human-like proportion and joints with wide movable ranges including two waist joints. The software segments the motion into the sequence of the contact states between the robot and the floor and assigns an appropriate controller to each transition between the consecutive states.

A humanoid having 12 DOF called RoboSapien was developed at National University of Singapore in 2003[20]. The gait is generated for a robot to walk on flat ground and climb up stairs. For the trajectory based gait generation, various parameters satisfy ZMP criterion and can realize continuous walking. The evolutionary algorithm is used to choose the parameters combinations which result in the best performance.

ASIMO humanoid is developed by Honda Motor Co.[21] ASIMO had 26 Degree of Freedom. 2 in the head, 5 for each Arms (3 DOF for shoulder joint, 1 at the elbow, and 1 at the wrist for each arm), 1 for each hand, and 6 for each leg (3 DOF for the hip joint, 1 for the knee joint and 2 for ankle joint). It uses enhanced visual and force sensor technologies for autonomous continuous movements and smoother interaction with people.

University of Queensland, Department of Information Technology and Electrical Engineering had developed a humanoid called GuRoo[22]. The goal of this project was to compete in RoboCup soccer competition. It has 23 DOFs , 1.2 meters tall and weighs 38 kilograms. The GuRoo uses eight RC servo motors to control the upper joints (head, neck and arms) and fifteen 70W DC geared motors. The lower DC motors are controlled by five identical controller boards, with each board controlling three motors each. The motors use velocities rather than angles for movements and control boards perform low-level control loops to maintain desired velocities. The desired velocity for each of the joints is calculated by a software on a PC or on a handheld device that allows the robot complete freedom.

1.5 Our Work

The approaches discussed so far require to solve dynamic equations or to calculate the inverse kinematics. Even the evolutionary approaches used so far can just focus on maximizing the distance covered by the robot or on controller tuning, but the evolution of human walking considers optimization of many parameters. So to expand the horizons of bipedal walking we are presenting a new evolutionary computation approach to obtain the stable walking of a biped. We are using a Multi-Objective Genetic Algorithm(MOGA) called Non-dominated Sorting Genetic Algorithm(NSGA-2)[14] to optimize the walking distance and the stability. In our work the controller of a robot is a fully-connected recurrent neural network whose weights are being optimized using NSGA-2. We have performed our experiments over six different types of robots having DOFs range from four to ten. One of the robots having six DOFs performs better than the rest because the figure six is good enough for free movement and small enough to reduce the search space for the evolutionary algorithm. Figure 1.4 shows the output of our work. The simulation has been carried out in a free library named **Open Dynamics Engine**[3] (ODE).

The Multi-Objective approach can lead to highly fast evolution of the bipedal walking with the increase in computing power of today's computers. One can also use this approach on application specific robots also. The crucial part in this approach is how to formulate your objective into the parameter to be optimized during the course of evolution. Let say, we can include Step length, Energy minimization, The morphology of the creature itself as an objective function in the Multi-Objective evolutionary algorithm. One can also make the robot to approximate the curve shaped trajectories also. Even the controller complexity can also be included in one of the objective functions of this algorithm. To make this approach working, computer simulations perform major role and hence computing power of the machines. Today simulating libraries are proven to be very good approximations of real robots

Simulation and robotics represent two different levels of complexity in modeling.

[3]http://www.q12.ode.org.

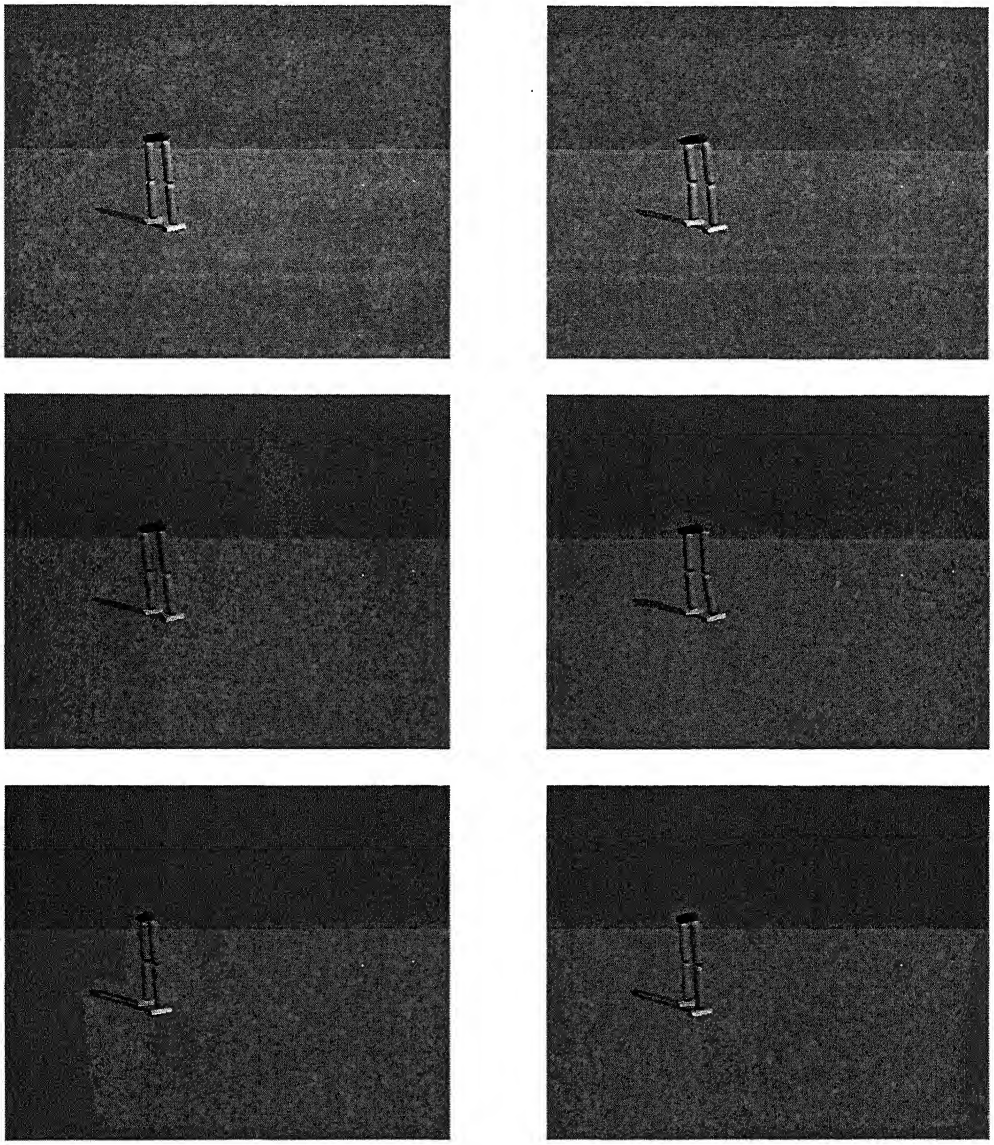


Figure 1.4: Snapshots from the simulation

Simulation models a simplified brain interacting with simplified (virtual) environment. Robotics models a simplified brain interacting with complex (real) environment. The target is to understand how complex human mind interacts with complex environment. When doing modeling, it would be better to make simulation first and then to build a robot. The reason is usually a robot is more expensive and needs more efforts to set up. Besides, it is more difficult to analyze enormous amount of stimulus from real world. We can still gain many insights by simulation if most of the important parameters have been chosen. In addition, we can also use simulation for a simple test of our model. If the test result is fine, then a robot is the next step to justify the model in the real world.

Chapter 2

Techniques/Robot/Simulation

2.1 Techniques

2.1.1 Neural Networks

Neural network-based control has been widely accepted by engineers and scientists in the control field due to the ability of neural networks to approximate arbitrary nonlinearities. In our work we have used general feed forward neural network for a standing robot and the recurrent neural network for a walking robot. Generally the training of the neural network is being done by a back propagation rule but here we are optimizing its weight using the evolutionary algorithms. For further information on Artificial Neural Networks refer [11]

2.1.2 Bezier Curves

A Bezier curve in its most common form is a simple cubic equation that can be used in any number of useful ways. Originally developed by Pierre Bezier in the 1970's for CAD/CAM operations.

Consider $N+1$ control points P_k ($k = 0$ to N) in 3 space. The Bezier parametric curve function is of the form...

$$B(u) = \sum_{k=0}^{N} P_k \frac{N!}{k!(N-k)!} u^k (1-u)^{N-k} \tag{1}$$

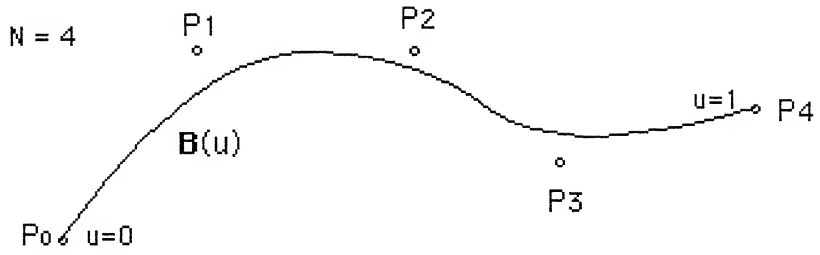


Figure 2.1: Bezier curve with 4 control points

$B(u)$ is a continuous function in 3 space defining the curve with N discrete ɔntrol points P_k. $u = 0$ at the first control point ($k = 0$) and $u = 1$ at the last ɔntrol point ($k = N$).

Notes:

- The curve in general does not pass through any of the control points except the first and last. From the formula $B(0) = P0$ and $B(1) = PN$.
- The curve is always contained within the convex hull of the control points, it never oscillates wildly away from the control points.
- If there is only one control point P_0, ie: $N = 0$ then $B(u) = P_0$ for all u.
- If there are only two control points P0 and P1, ie: N=1 then the formula reduces to a line segment between the two control points.

$$B(u) = \sum_{k=0}^{1} P_k \frac{N!}{k!(N-k)!} u^k (1-u)^{N-k} = P_0 + u(P_1 - P_0) \tag{2}$$

- The term

$$\frac{N!}{k!(N-k)!} u^k (1-u)^{N-k} \tag{3}$$

is called a blending function since it blends the control points to form the Bézier curve.

- The blending function is always a polynomial one degree less than the number of control points. Thus 3 control points results in a parabola, 4 control points a cubic curve etc.
- Adding multiple control points at a single position in space will add more weight to that point 'pulling' the Bézier curve towards it.

1.3 Multi-Objective Genetic Algorithm

ver the past decade, a number of multi-objective evolutionary algorithms (MOEAs) ve been suggested [14]

The primary reason for this is their ability to find multiple Pareto-optimal solu-ns in a single simulation. Since the principal reason why a problem has a multi-jective formulation is because it is not possible to have a single solution which nultaneously optimizes all objectives, an algorithm that gives a large number of ternative solutions lying on or near the Pareto-optimal front is of great practical lue. In the following, we discuss one such multi-objective EA – Non-dominated rting genetic algorithm (NSGA-II), which is used in our work.

NSGA-II Algorithm: The main loop

itially, a random parent population P_0 is created. The population is sorted ased on the non-domination. Each solution is assigned a fitness equal to its non-mination level (1 is the best level). Thus, minimization of fitness is assumed. nary tournament selection, recombination, and mutation operators are used to eate a child population Q_0 of size N. From the first generation onward, the pro-dure is different. The elitism procedure for $t \geq 1$ and for a particular generation shown in the following:

$= P_t \cup Q_t$ // combine parent and children population
$=$ `fast-nondominated-sort`(R_t)
$\mathcal{F} = (\mathcal{F}_1, \mathcal{F}_2, \ldots)$, all non-dominated fronts of R_t
$_{+1} = \emptyset$

ıntil $|P_{t+1}| < N$ // till the parent population is filled
`crowding-distance-assignment(`$\mathcal{F}_i$`)`
'/calculate crowding distance in $\mathcal{F}_i$
$P_{t+1} = P_{t+1} \cup \mathcal{F}_i$ //include i-th non-dominated front in the parent pop
)ort$(P_{t+1}, \prec_n)$ //sort in descending order using $\prec_n$
$^{2}_{t+1} = P_{t+1}[0 : N]$ //choose the first N elements of P_{t+1}
$\mathcal{Q}_{t+1}$ = `make-new-pop(`P_{t+1}`)`
'/use selection, crossover and mutation to create a new population Q_{t+1}
$= t + 1$

First, a combined population $R_t = P_t \cup Q_t$ is formed. The population R_t will ɔe of size $2N$. Then, the population R_t is sorted according to non-domination. The ıew parent population P_{t+1} is formed by adding solutions from the first front till :he size exceeds N. Thereafter, the solutions of the last accepted front are sorted ıccording to $\prec_n$ and a total of N solutions are picked. This is how we construct the ɔopulation P_{t+1}. This population of size N is now used for selection, crossover and nutation to create a new population Q_{t+1} of size N. It is important to note that we ıse a binary tournament selection operator but the selection criterion is now based ɔn the niched comparison operator $\prec_n$.

A Fast Non-dominated Sorting Procedure

In this approach, every solution in a GA population is checked with a partially filled population for domination. To start with, the first solution from the population is kept in a set P'. Thereafter, each solution p (the second solution onwards) is compared with all members of the set P' one by one. If the solution p dominates any member q of P', then solution q is removed from P'. This way non-members of the non-dominated from get deleted from P'. Otherwise, if solution p is dominated by any member of P', the solution p is ignored. If solution p is not dominated by any member of P', it is entered in P'. This is how the set P' grows with non-dominated solutions. When all solutions of the population is checked, the remaining members of P' constitute the non-dominated set.

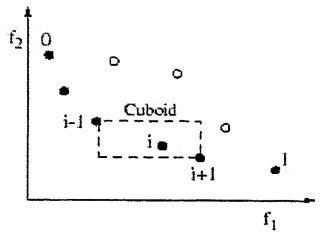


Figure 2.2: Crowding distance

`fast-nondominated-sort(`P`)`
$P' = \{1\}$ //include first member in P'
for each $p \in P \wedge p \notin P'$ //take one solution at a time
$P' = P' \cup \{p\}$ //include p in P' temporarily
for each $q \in P' \wedge q \neq p$ //compare p with other members of P'
if $p \prec q$, then $P' = P' \backslash \{q\}$
//if p dominates a member of P', delete it
else if $q \prec p$, then $P' = P' \backslash \{p\}$
//if p is dominated by some $z \epsilon P'$, don't include p in P'

To find other fronts, the members of P' will be discounted and the above procedure is repeated.

∎ *Density estimation*

To get an estimate of the density of solutions surrounding a particular point in the population we take the average distance of the two points on either side of this point along each of the objectives. This quantity $i_{distance}$ serves as an estimate of the size of the largest cuboid enclosing the point i without including any other point in the population (we call this the *crowding distance*). In Figure 2.2, the crowding distance of the i^{th} solution in its front (marked with solid circles) is the average side-length of the cuboid (shown with a dashed box).

The following algorithm is used to calculate the crowding distance of each point

in the set $\mathcal{I}$:

```
crowding-distance-assignment(I)
```
$l = |\mathcal{I}|$ //number of solutions in $\mathcal{I}$
for each i, set $\mathcal{I}[i]_{distance} = 0$ //initialize distance
for each objective m
  $\mathcal{I} = \text{sort}(\mathcal{I}, m)$ //sort using each objective value
  $\mathcal{I}[1]_{distance} = \mathcal{I}[l]_{distance} = \infty$ //so that boundary points are always selected
  for $i = 2$ to $(l - 1)$ //for all other points
    $\mathcal{I}[i]_{distance} = \mathcal{I}[i]_{distance} + (\mathcal{I}[i + 1].m - \mathcal{I}[i - 1].m)$

Here $\mathcal{I}[i].m$ refers to the m-th objective function value of the i-th individual in the set $\mathcal{I}$. The complexity of this procedure is governed by the sorting algorithm. In the worst case (when all solutions are in one front), the sorting requires $O(mN \log N)$ computations.

Crowded comparison operator

The crowded comparison operator ($\prec_n$) guides the selection process at the various stages of the algorithm towards a uniformly spread-out Pareto-optimal front. Let us assume that every individual i in the population has two attributes.

1. Non-domination rank (i_{rank})
2. Local crowding distance ($i_{distance}$)

We now define a partial order $\prec_n$ as :

$i \prec_n j$ if ($i_{rank} < j_{rank}$) or (($i_{rank} = j_{rank}$) and ($i_{distance} > j_{distance}$))

That is, between two solutions with differing non-domination ranks we prefer the point with the lower rank. Otherwise, if both the points belong to the same front then we prefer the point which is located in a region with lesser number of points (the size of the cuboid inclosing it is larger).

Let us now look at the complexity of one iteration of the entire algorithm. The basic operations being performed and the worst case complexities associated with are as follows:

1. Non-dominated sort is $O(mN^2)$,

2. Crowding distance assignment is $O(mN \log N)$, and

3. Sort on $\prec_n$ is $O(2N \log(2N))$.

As can be seen, the worst-case complexity of the above algorithm is $O(mN^2)$. where N is the population size and the number of objectives is m. At most N points along the pareto-optimal front are identified by the algorithm so the complexity of the algorithm is directly a function of the accuracy to which the pareto-optimal front is to be identified (N).

2.2 Simulation

Our simulation must include the laws of physic to approximate the reality. But physics is a huge field, and of course only a small part of it is needed to make a robot walk. So, let's see exactly what we need and how it can be simulated.

From the physics point of view, a robot is simply an assembly of many rigid bodies connected together with some joints. Each one of these bodies can interact with its neighbors: a force or a torque applied on one body will also affect its connected neighbors, and all the bodies must be able to bounce into each other. Finally all the bodies have to be affected by a force of gravitation, otherwise the robot couldn't fall. In others words, we need a tool for simulating articulated rigid body dynamics. One could easily imagine that creating such a tool is already a huge project in itself. Hopefully there are many fields in which rigid body simulations are required. For example, the video games industry is very interested in this technology to make theirs games more and more realistic.

In this work we use the free and open source library called ODE, elaborated in the next section.

2.2.1 ODE

ODE stands for Open Dynamic Engine. ODE is an open source, high performance library for simulating rigid body dynamics. It is fully featured, stable, mature and platform independent with an easy to use C/C++ API. It has advanced joint types and integrated collision detection with friction. ODE is useful for simulating vehicles, objects in virtual reality environments and virtual creatures. It is currently used in many computer games, 3D authoring tools and simulation tools.

In ODE first we have to create a world of objects. Then we have to create the bodies in this world(enviroment) and we can associate mass and the geometric object of specific shape to this body. ODE supports objects having very primitive shapes (Sphere, Box, Capped Cylinder). Now one can set the initial position, orientation and initial velocity of each of the bodies. The position is specified into world coordinate systems and orientation should be specified with respect to body frame i.e. the angle of rotation around each of the axes. After placing the bodies at the specific positions in the world, Joints comes into picture. Joints can be placed only between two bodies. The user have to specify the type of joint, the axes and the anchor point. ODE supports seven types of joints.

- Ball and Socket
- Hinge
- Universal
- Hinge-2
- Slider
- Fixed
- Contact

ODE creates temporary contact joint between colliding bodies, Bodies attached by fixed joints would behave like a single body.

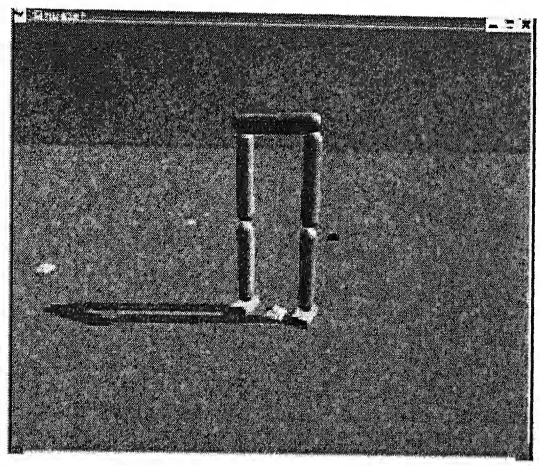


Figure 2.3: First Model in ODE

After creating bodies/joints, the world is ready to step into simulation process. User have to specify the time step for the simulation, The time step is the time duration after which the position and orientation of bodies is being calculated. One has to apply the forces and/or torques after each time step. ODE also does the collision detection between the objects after each time step. It calls a special function for collision detection where user can recognize the two bodies being collided and can set the surface parameters. If the user wants to display the simulation then he has to get the position and orientation of the objects and have to draw the shapes using a graphics library, say OpenGL. User can set the camera positions/orientation and can set the terminating conditions also.

- *Simple articulated model in ODE*

Figure 2.3 shows the first articulated model of the robot, elaborated in next section.

2.3 The Robot itself

The robot is a simple biped structure with seven links(waist,upper legs, lower legs and feet) without an upper torso. We have performed our experiments on the robots having different DOFs(4,6,8,10) that can be made using such seven links structure.

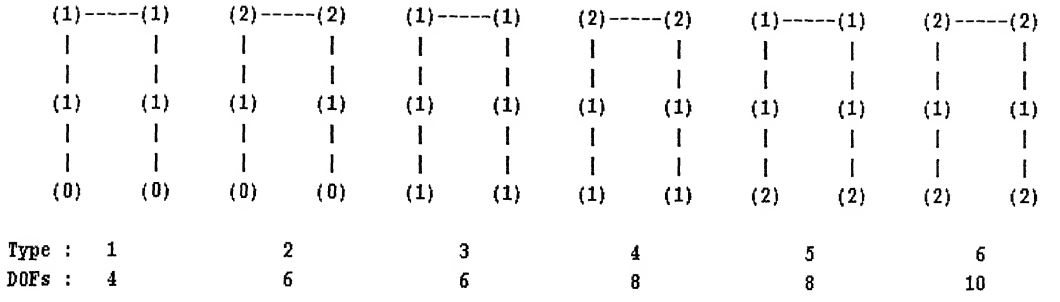


Figure 2.4: The types of Robot used in Experiments

Index	*Object*	*Shape*	*Dimensions*	*Mass*
1	Foot	Box	l=10cm, w=5cm, h=2.5cm	1 kg
2	Lower Leg	Capped Cylinder	l=20cm,r=2.5cm	1 kg
3	Upper Leg	Capped Cylinder	l=20cm,r=2.5cm	1 kg
4	Waist	Capped Cylinder	l=20cm,r=2.5cm	1 kg

Table 2.1: Properties of elementary objects used in creating the robot

Figure 2.4 shows the types of robot used in our experiments. The robot of *Type 1* is having four DOFs, *Type 2* and *Type 3* have six DOFs, *Type 4* and *Type 5* have eight DOFs and *Type 6* has ten DOFs. In the figure 2.4, the digit in the paranthesis indicates the number of DOF at that particular joint. For any joint if the DOF is 1 then it is in the saggital plane and if it has 2 DOFs then one in the saggital plane and one in the frontal plane. The DOF in saggital plane is known as pitch joint and the same in the frontal plane is known as roll joint. The digit 0 indicates fixed joint and hence zero DOF at that joint.

The morphological parameters of one such robot(Figure 2.3) are given in TABLE 2.1. The joint constrains are depicted in TABLE 2.2.

Index	*Joint*	*Plane of Rotation*	*Range of Motion*
1	Knee	Saggital	0 to $\frac{\pi}{2}$
2	Hip	Saggital	$\frac{-\pi}{4}$ to $\frac{\pi}{4}$
3	Hip	Frontal	$\frac{-\pi}{4}$ to $\frac{\pi}{4}$
4	Ankle	Saggital	$\frac{-\pi}{6}$ to $\frac{\pi}{6}$
5	Ankle	Frontal	$\frac{-\pi}{6}$ to $\frac{\pi}{6}$

Table 2.2: Joint Limits

Chapter 3

Our Approaches and Results

3.1 Learning to stand

The first milestone of this work was to make the robot standing. If the applied technique is not able to make the robot standing on his two feet then it is almost impossible to get him walking by applying the same techniques. The Morphology of a robot is as described above and the robot used for this experiment has six degrees of freedom. Robot of **Type 2** from the Figure 2.4. We have tried very primitive versions of both the techniques Artificial neural network and Genetic Algorithms to get him in a standing stance.

3.1.1 The Neural Controller

The neural network used to achieve the standing robot is simple feed forward network as shown in Figure 3.1. The architecture of the network comprises of three layers- Input Layer, Hidden Layer, Output Layer. The input layer has six neurons in it, one for each of the joint angles, also one bias neurons which is fully connected to the hidden layer and emits the signal of +1 always. The hidden layer consists of 12 neurons and output layer has six neurons, one for the torque value to be applied to each of the joints.

The activations of the hidden and output neurons are computed by

$$a = \sum_{i=1}^{n} W_{ij} O_i \tag{1}$$

Where O_i is the output of a neuron in the previous layer. In case of the hidden layer of the Recurrent Neural Networks(discussed later), O_i represents both the output from the input layer at the current time step, and the output from the hidden layer at the previous time step. W_{ij} is the weight of the synapse connecting them. The output of neurons in the hidden and output layer is given by

$$f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{2}$$

Above function is also known as *tansig* function. The torque values applied to the joints can be negative and positive both and the *tansig* function ranges from -1 to 1 and it is contiguous so it was obvious choice as an activation function. The output is scaled from the range [-1,1] to the range [-MaxT,MaxT] where MaxT is the maximum torque required for any of those six joints at any instance of time to prevent the robot from falling down. The torques, obtained through such method are being applied to the respective joints of the robot which try to make the robot not to fall for very long time. This leads the robot to stay in standing posture mostly.

3.1.2 Genetic Algorithms

In most of the cases the training of neural network is being done by backpropagation of an error, but this technique requires the set of training data which is not available in our case, so the obvious choice for getting desired weights of the links of the neural network is Genetic Algorithms, we call it optimization of weights of neural network rather than training. The weights of the neural network range from -1 to 1. We have used Binary-coded Genetic Algorithms which is the most commonly used and primitive version of GA. 14 bits are used to represent a single weight and hence 144 x 14= 2016 bits required to represent one gene corresponding to a single neural controller. The crossover probability used is 80% and mutation probability is 10%. Conventional one-point crossover operator was used here. This simple

genetic algorithm uses roulette wheel selection method. Roulette wheel selection says that the probability of a gene being selected for the next generation is directly proportional to its fitness. Here in this case the fitness is the time for which a robot can remain stable. The population size used here is 200. A robot that can stand more than 2000 time steps of the simulation then it is declared as a stable one and GA terminates the simulation of this robot and moves forward to evaluate the next gene(robot), Termination of such robot is required because if such a robot will never fall down then simulation will never turn off and GA will not move forward. Also we are saving the robot(i.e. gene) after evaluating its behavior so if we don't put such a termination condition we cannot save a stable robot. Figure 3.2 shows the convergence graph of Genetic Algorithm. After 20 generations GA converges and most of the robots in the population can stand. Here the robot has got a standing posture exactly the same way a human stands i.e. by constantly applying very small amount of torques at each of its joints. As you can see in the Figure 3.3 the angles of the knee joints and the hip roll joints are almost zero most of the times. Torques applied to each of the joints remain constant as it is demanded for a standing posture.

The graphs have been plotted for first 500 time steps of the simulations. The red plots are those of joint angles and the blue ones are those of torques applied. The vertical axis indicating joint angles has the range from $-\pi$ to π and that of torque is from -3Nm to 3Nm.

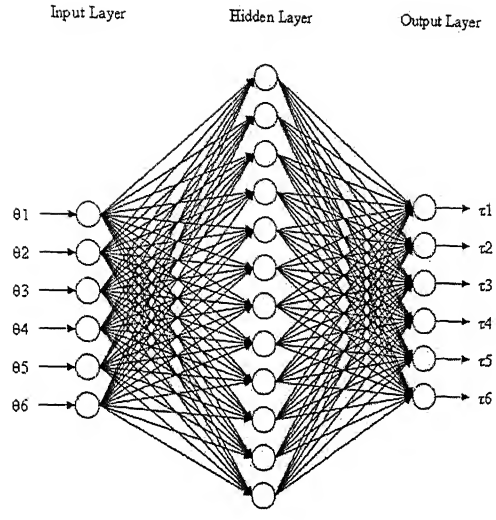


Figure 3.1: Neural Controller for a Standing Robot

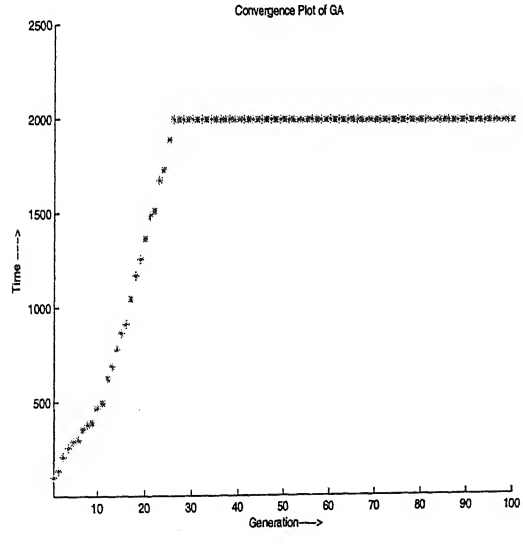
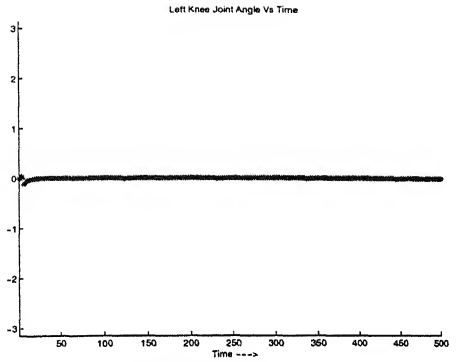
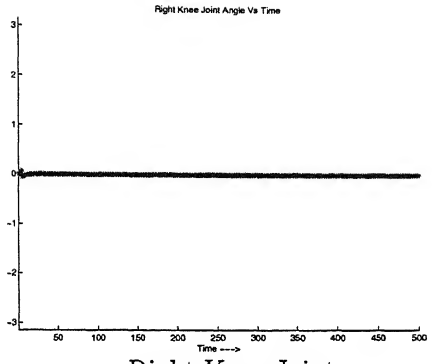
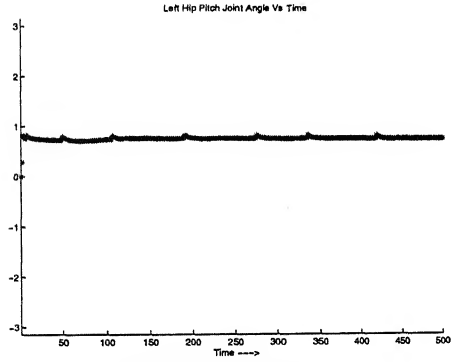
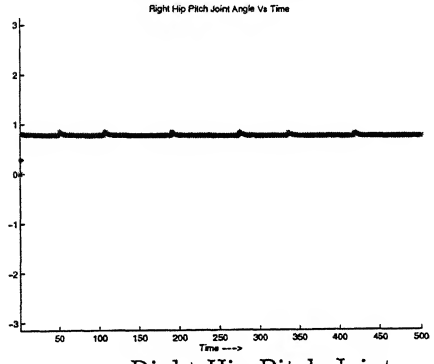


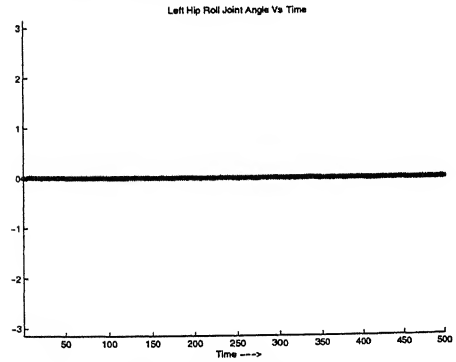
Figure 3.2: Convergence Plot of GA that searches for a standing robot











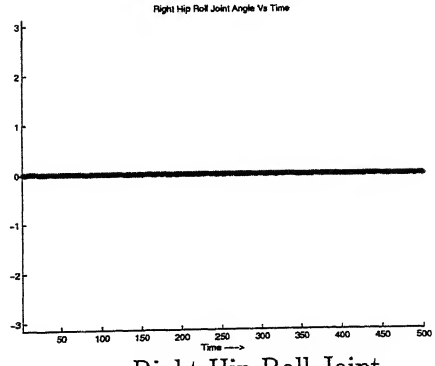
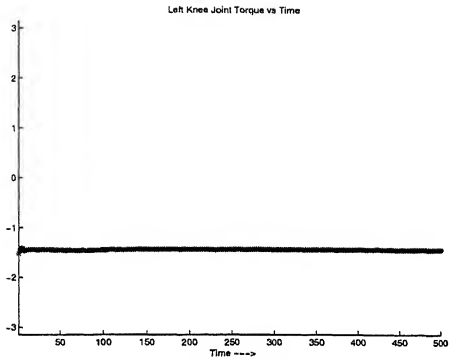
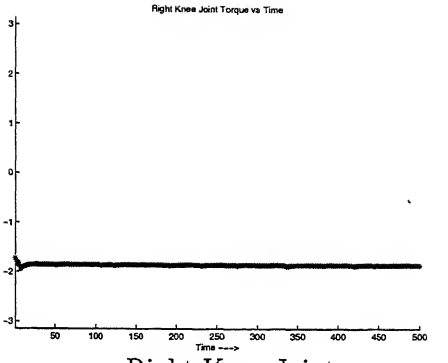
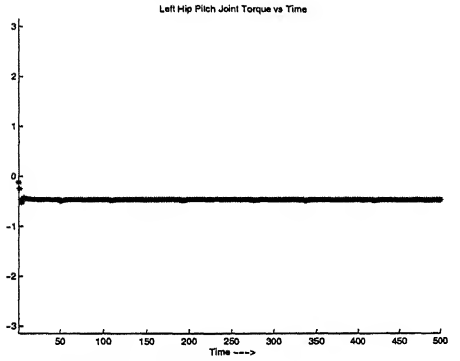
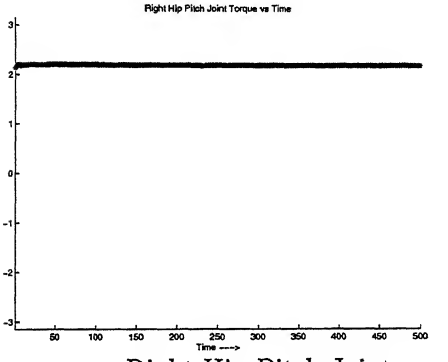


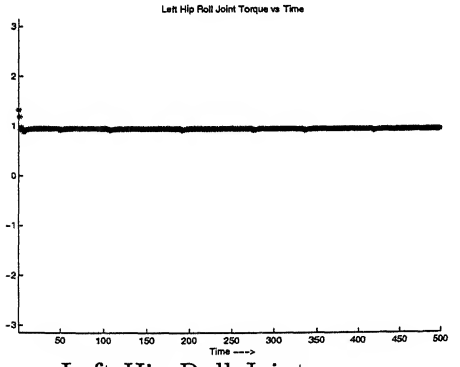
Figure 3.3: Joint Angles(in radians) for a standing robot











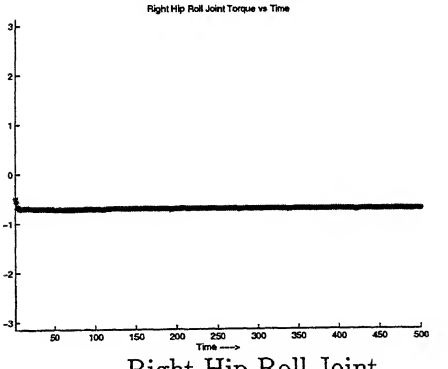


Figure 3.4: Joint Torques(in Nm) for a stading robot

3.2 Learning to walk

The major goal of this project is to make the biped robot walking autonomously. The majority of robots that can roam around autonomously are mobile robots with wheels and have four or more contacts with the ground at all the times results in static stability.

Biped locomotion is inherently unstable and presents a more difficult task. Adding more number of DOFs, Torso, arms and head compounds the problem by raising the center of mass and creating more joints that need controlling in order to walk the robot. These parameters also increases the search space with for an evolutionary algorithm. The following sections discuss the controllers(Recurrent Neural Network and Bezier Curves) used to get a walking robot and the evolutionary algorithm that is used to evolve such controllers. We have tried both the controllers for the robots having DOFs range from 4 to 10. The following section describes the results obtained with a robot of **Type3** with the required plots.

3.2.1 The Neural Controller

Most of the times Feed Forward Neural Network is used for pattern classification or to make the next move in the board games. In this kind of applications ANN just takes the current state of the board game and throws next move as an output or will take some pattern (say a picture) to be classified and throws the class(for an instance 'smiling' or 'frowning' face in case of picture classification) as an output in which it falls. In all such cases the output of the network doesn't depend on the previous outputs.

Walking is a periodic task and also the next action to be performed(torque applied at any particular joint) depends on the past events(i.e. *past values of torques* that determines current posture of the robot). One should also keep track of whole posture of the robot before applying torque to any joint that depends on the torques applied previously. Therefore simple feed forward networks are not suitable for this

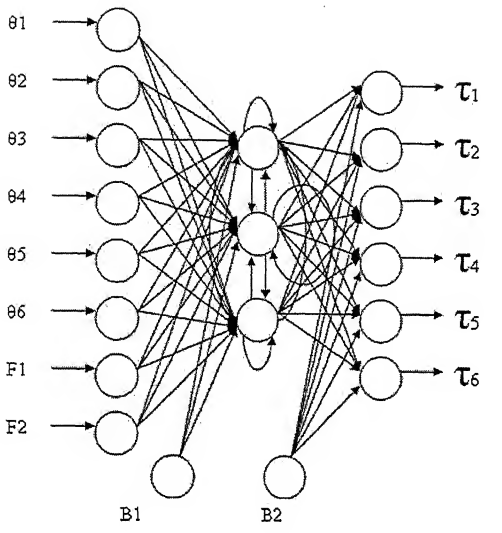


Figure 3.5: The Neural Controller for a Walking Robot

purpose. We have tried our simple feed forward network and tried to evolve with the GA but even after more than 200 generations not a single robot could walk a single step. For this kind of applications fully connected Recurrent Neural Network is a better choice, in this type of networks the previous output is being maintained and will be fed as an input in the current time step.

Figure 3.5 depicts such a fully connected Recurrent Neural Network that is used as a controller for a walking robot. The input layer is having eight neurons. Six Neurons from the input layer takes joint angles of each of the joint as an input and two more neurons for each of the foot sensors. Neuron corresponding to Foot sensor emits the binary signal one or zero having foot in contact with the ground and having the foot in the air respectively. The hidden layer consists three neurons. We have kept the number of neurons in the hidden layer a variable component so the given figure of a network is just a sample. The output layer has six neurons. There stands one neuron for the torque to be applied to each of the joints and hence six in total. B1 and B2 in the Fig3.5 are bias neurons which emit a signal of one every time.

As described previously, The activation function of the neurons in the hidden layer and in the output layer is a *tansig* function. The output of the hidden layer in previous time step is being fed as in input to hidden layer and output layer in the current time step. The *tansig* function ranges from -1 to 1 and hence the range of the outputs, which has been scaled to the range, -MaxT to +MaxT . Here we have chosen 12Nm as MaxT. This value is big enough to cover the maximum torque required for proper walking and at the same time avoids a large search space for GA. The weight of the links of the network range between -1 to 1 which are being optimized using a multi-objective genetic algorithm called NSGA-2, discussed in the subsequent section.

Note: This Architecture of Neural Controller is for the Robot having six DOFs. We have performed our experiments for six different types of robots as described in chapter 3

3.2.2 The Multi-Objective Genetic Algorithm

We have tried to make the robot walking using recurrent neural network with Simple Genetic Algorithms, here the fitness is the horizontal distance traveled by a robot. The results were not at positive side with this approach. The major problem was that the robot starts falling down because the structure of a robot used is not statically stable one and hence when GA forces it to move forward, it falls down. This implies that the torque values required for initial time steps of the simulation are such that they make the robot standing for a while and then these values should approach towards the way that can make the robot move forward. Hence we need to optimize two parameters the time(the one we used for a standing robot) and the distance covered in horizontal direction. To optimize more than one parameters we have used MOGA called NSGA-2 as described previously.

The population size chosen is 200 and each gene contains (27 + 9 + 24) 60 real variables corresponding to each of the links in the Neural Controller. The two objective parameters to be optimized are *Horizontal Distance* and *Time*. The goal

is to cover maximum distance but inclusion of time maximization helps the robot to remain stable. Robot starts walking from a standing position so considering time as one of the optimization parameters, prevents the robot from falling down. As shown in the Figure 3.6 MOGA converges after 200 generations. The plot depicts the fittest of each generation and the fitness is the distance in the centimeters. The fittest robot walks more than 25 meters distance. Now because of the second optimization parameter *Time*, the speed of the robot is not improving. As pointed out earlier the optimization parameter *Time* is just to make the robot stable for first few steps so we can exclude it after some time and only *Horizontal Distance* would remain as an optimization parameter. This can be done by keeping one threshold value of time τ beyond which the value of the second optimization parameter of a gene would be τ only instead of the real time it has taken. So MOGA would automatically put a stress in maximizing the distance and hence increases the speed, Technically speaking here we are putting a hyperplane in the search path of the MOGA. The fittest robot with this approach could walk more than 150 meters of distance as shown in Figure 3.7. The gait of the robot is not as good as humans but looks quite realistic, it hardly bends the knees which is clearly visible in the Figure 3.8 (Knee angle remains almost zero most of the times.) which leads to a shuffling motion. We have also tried only *Time* as the objective function and we have got the standing robot. But keeping only *Horizontal Distance* as an objective parameter robot falls down quickly and it doesn't learn to walk anymore even up to 500 generations. So the results truly demonstrates the trade-off between *stability* and *distance*. Figure 3.10 show the paths of a robot which depicts that the robot is learning to walk over the generations. From these figures it is very clear that the *Horizontal Distance* covered by robot is increasing over the generations.

We have tried our approaches for the robots having 4 DOFs, 6 DOFs, 8 DOFs and 10 DOFs. Figure 3.11 shows the paths of such robots. The robot having 10 DOFs couldn't walk even few steps because of increasing DOF increases the number of links in the neural controller and hence the search space for the MOGA. Other types of robot could walk up to good amount of distance. The robot of *Type3* performs well over the others. The roll joint at the ankle is very crucial for shifting the CG

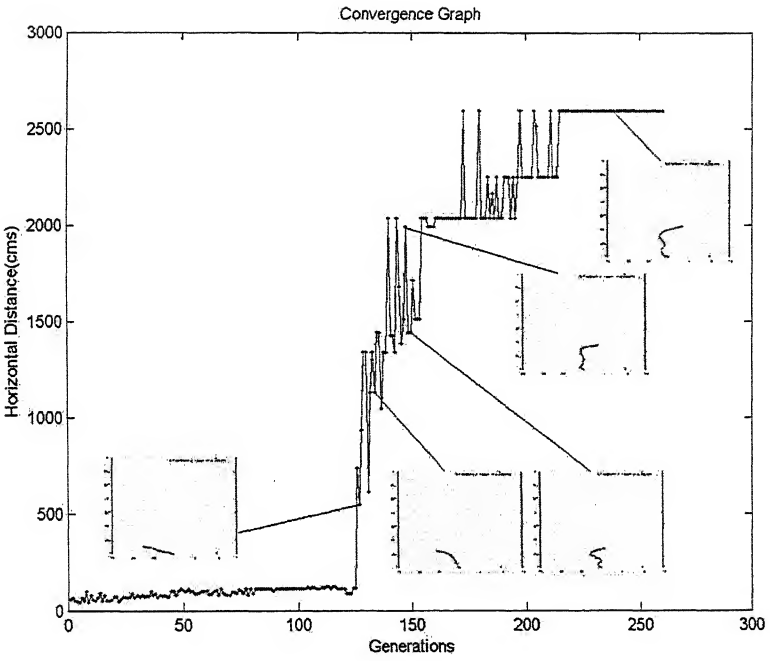


Figure 3.6: Convergence graph with two optimization parameters:Distance and Time

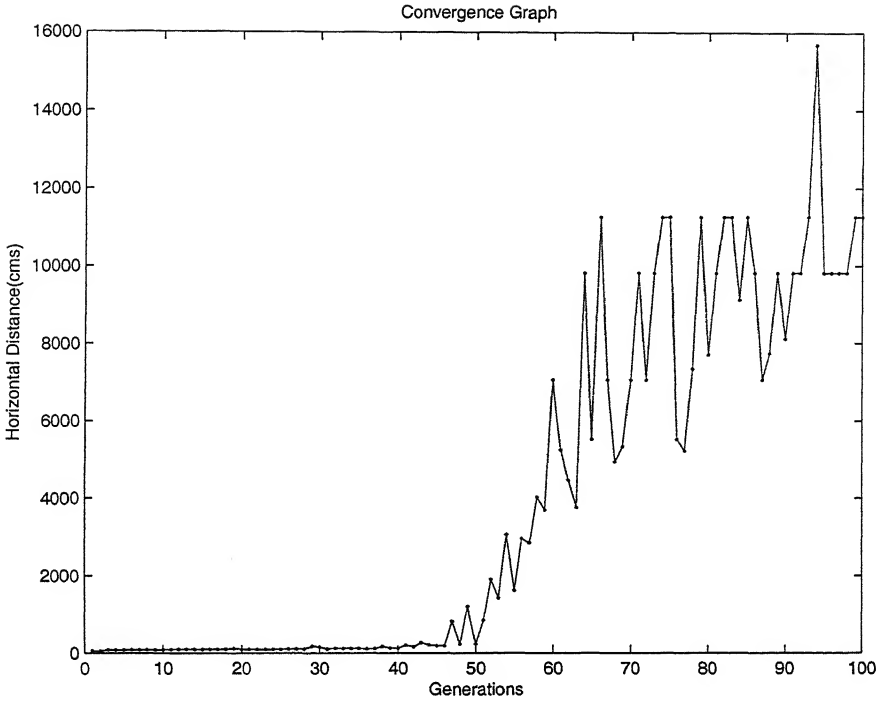


Figure 3.7: Convergence graph with one optimization parameter:Distance, After removing *Time* from the optimization parameters after few time steps

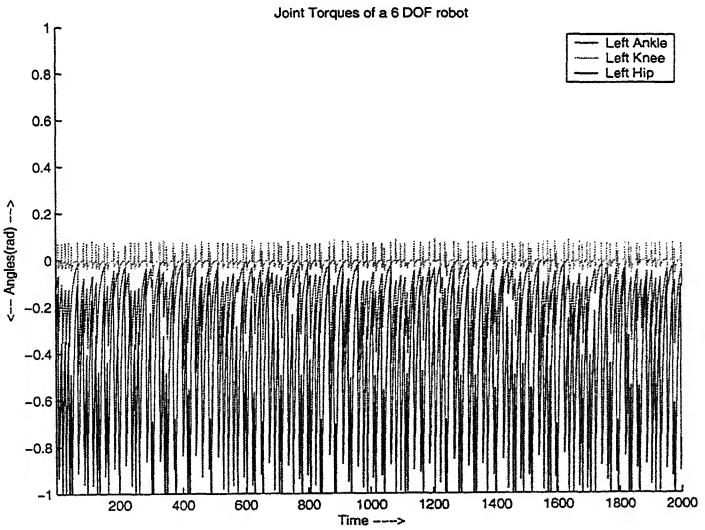


Figure 3.8: Joint Angles(radians) over Time

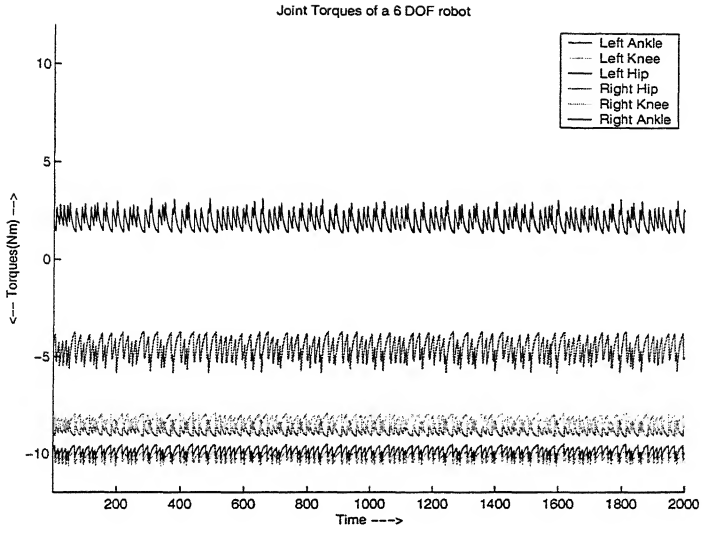
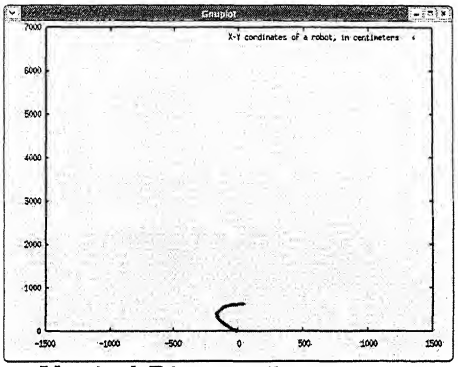
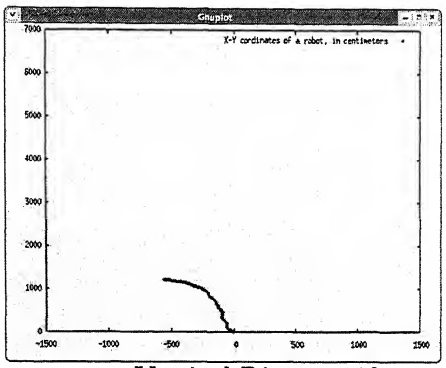


Figure 3.9: Torques(Nm) over Time

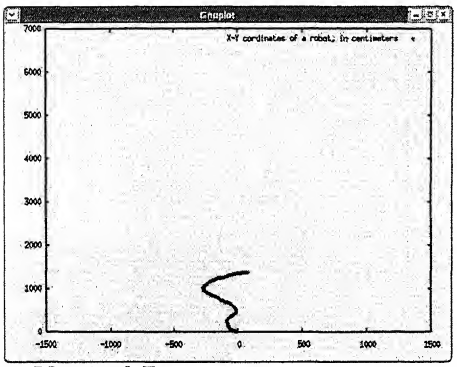
towards the resting leg and lifting the other one. Moreover increasing the DOFs demands increase in number of hidden neurons for better performance, as explained in [23], mind complexity is directly proportional to morphological complexity of a creature. We have also done our experiments with the robot having upper torso. Torso raises the center of mass and hence the difficulty in walking. Our robot with torso keeps roaming around the initial position as he has to refrain from falling down and have to be busy in balancing himself. As we increase the number of degrees of freedom then the robot becomes more unstable and it was falling down before taking initiative steps for walking so for the robots having higher degrees of freedom we have kept the larger foot size. This larger foot size helps the robot in maintaining the ZMP(Zero Moment Point) inside the foot area.



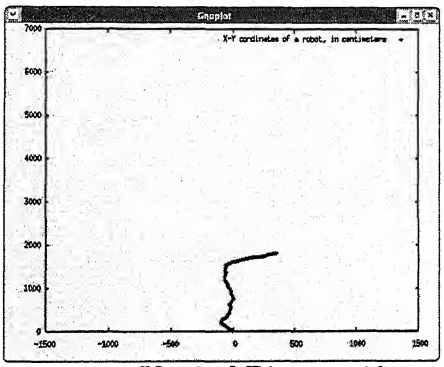
Vertical Distance:5m



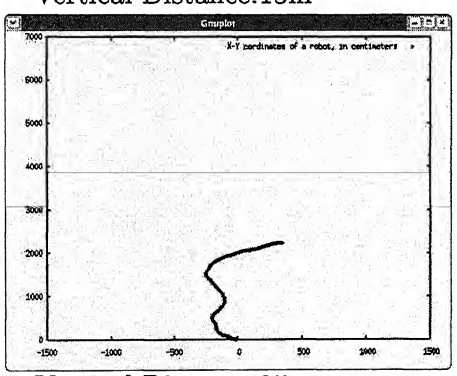
Vertical Distance:12m



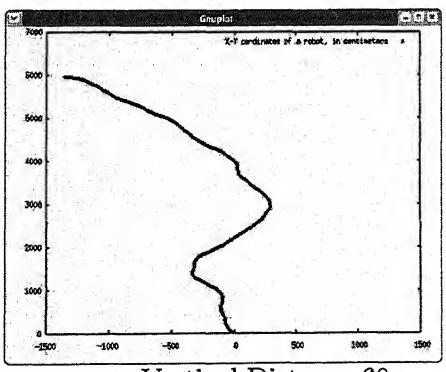
Vertical Distance:15m



Vertical Distance:18m

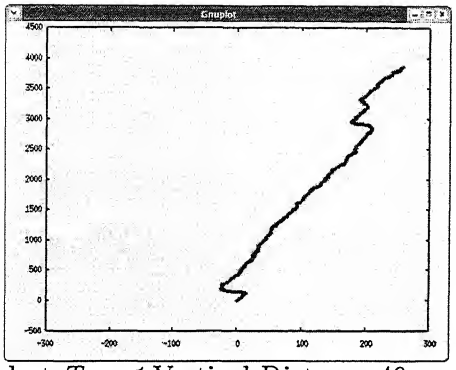


Vertical Distance:25m

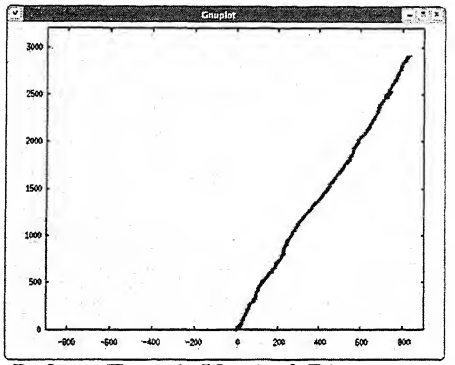


Vertical Distance:60m

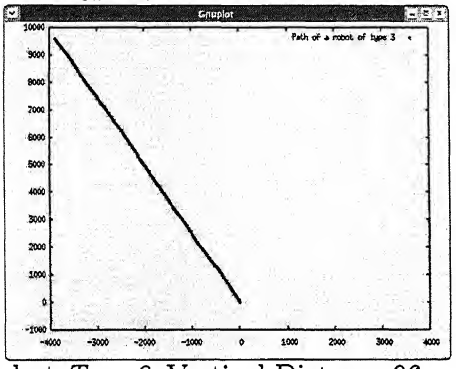
Figure 3.10: Paths of a robot over subsequent generations



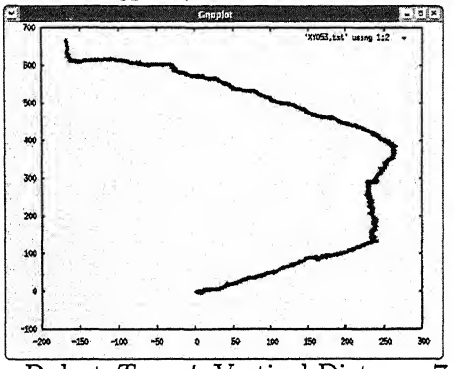
Robot *Type 1*,Vertical Distance:40m



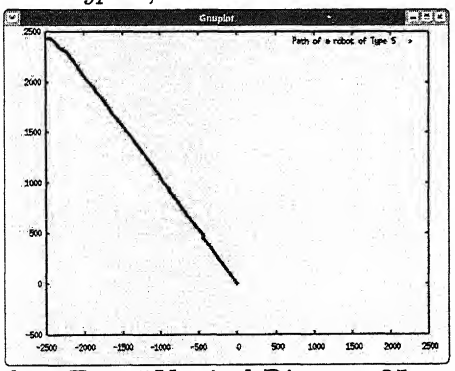
Robot *Type 2*, Vertical Distance:30m



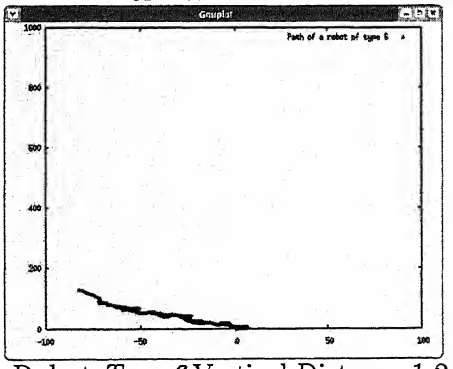
Robot *Type 3*, Vertical Distance:96m



Robot *Type 4*, Vertical Distance:7m



Robot *Type 5*,Vertical Distance:25m



Robot *Type 6*,Vertical Distance:1.2m

Figure 3.11: Paths of different types of robots

3.2.3 Bezier Curves in the Controller

The bipedal walking is very smooth gesture, synonymously speaking there are no jerks during the movements. This implies that the torques being applies at the joints are in the manner of either continuously increasing or decreasing. To model such continuous curves we have used Bezier Curves. We have assumed here that torques applied to each joints are Bezier Curves of four control points. As described in the previous chapter we can get the points on the curve by putting the value of the parameter u from 0 to 1. For $u = 0$ the point is the first control point itself and for $u = 1$ it is the fourth control point. The curve always remains in the convex hull obtained with four control points. We have fixed the range of the values of the control points from -MaxT to MaxT and MOGA will search the desired values of this control points in this range. The choice 'four' for the number of control points is good enough to mimic smooth and continuous curves and small enough to reduce the search space for MOGA. Figure 3.12 shows the convergence graph of a MOGA applied to optimize such torque curves. This plot is obtained by performing experiments on the robot of *Type 3*. For such a six DOFs robot there are $6 \times 4 = 24$ variables in a single gene of MOGA. Figure 3.13 shows the path of the one of the fittest robot achieved using this approach. Figure 3.14 shows the torques(in the shape of Bezier Curves) of the fittest individual obtained using this approach. Figure 3.15 portrays the joint angles for such a fittest robot.

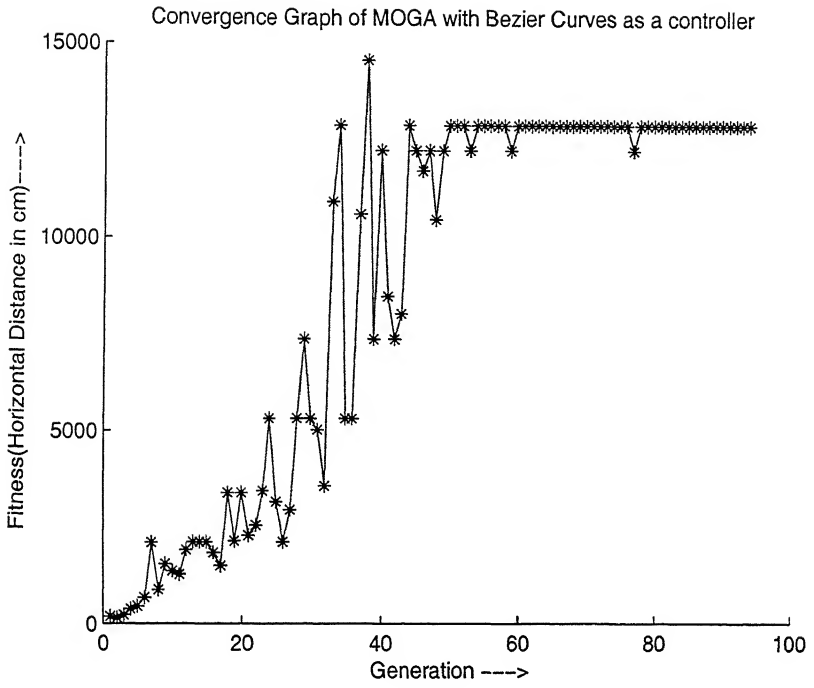


Figure 3.12: Convergence Graph of MOGA optimizing Bezier Curves

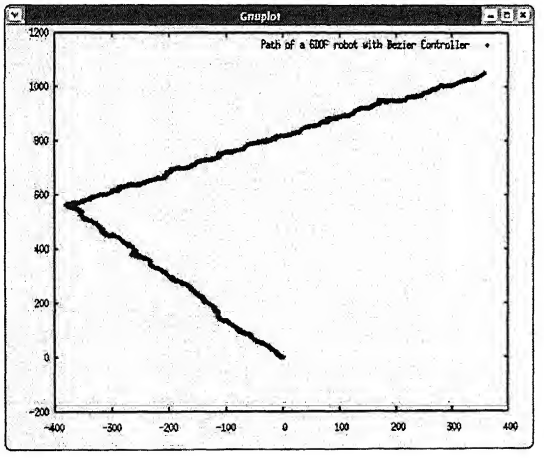


Figure 3.13: Walking path of a robot having Bezier Controller

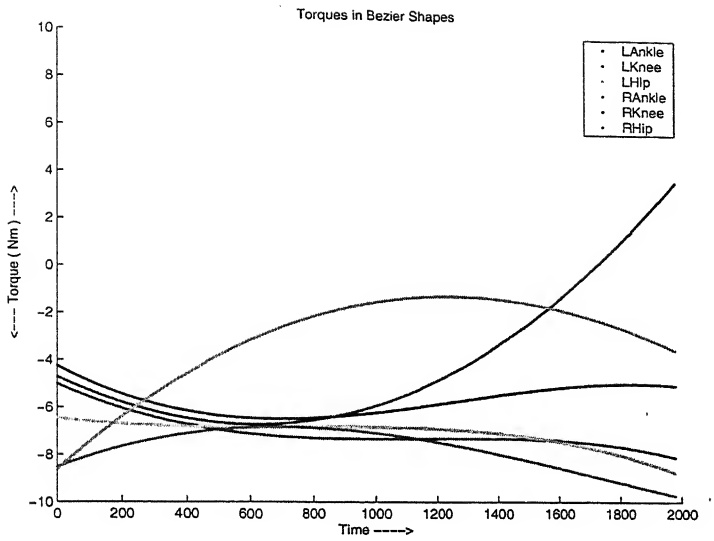


Figure 3.14: Torques of the fittest using Bezier Approach

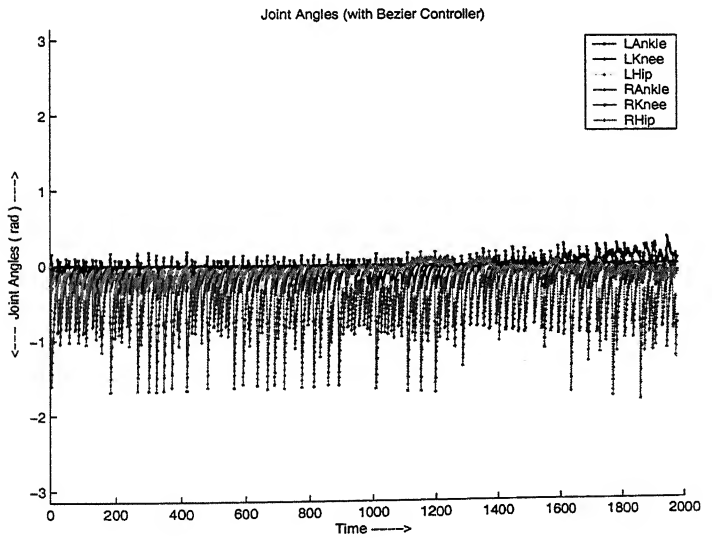


Figure 3.15: Joint Angles of the fittest using Bezier Approach

Chapter 4

Conclusion and Future Work

We have demonstrated a multi-objective approach to evolve artificial neural networks for controlling the locomotion of a 3D, physically simulated artificial biped creature. The pareto-frontier that resulted from each single run provided a set of ANNs which maximized locomotion capabilities of creature and also improved the stability in the initial phase of walking(which is very crucial). For further enhancement of this work there are plenty of other parameters one can include in the objective functions in MOGA like

- The complexity of the network i.e. Number of Hidden Neurons.
- Morphology of the robot
- Step Length
- Energy minimization

Bibliography

[1] Mattias Wahde and Jimmy Pettersson. A brief review of bipedal robotics research, 2002.

[2] Christine Azevedo, Philippe Poignet, and Bernard Espiau. Artificial locomotion control: from human to robots. In *Robotics and Autonomous Systems*, volume 47, pages 203–223, 2004.

[3] Prasad KulKarni. Dynamic analysis of biped robot locomotion and design, development, experimentation of a statically stable biped robot, 2004.

[4] Richard Jones. Walking like a two year old : A robot simulation. In *MEng Computer Science and Cybernetics*, 2003.

[5] Hamid Benbrahim. Biped dynamic walking using reinforcement learning, 1996.

[6] Jonathan Roberts, Damien Kee, and Gordon Wyeth. Improved joint control using a genetic algorithm for a humanoid robot, 2001.

[7] Ken Endo, Takashi Maeno, and Hiroaki Kitano. Acquisition of humanoid walking motion using genetic algorithm-considering characterstics of servo modules-. In *International Conference on Robotics and Automation Washinton, DC May 2002*, 2002.

[8] Abraham L Howell. Robot control using genetic prograaming, 2003.

[9] John Koza. *Genetic Programming.* The MIT Press, 1992.

[10] G. M. Maxwell D. McMinn and C. MacLeod. An evolutionary artificial nervous system for animat locomotion, 1998.

[11] Martin Hagan, Howard Demuth, and Mark Beale. *Neural Network Designs.* Thomson Learning, 2002.

[12] Ken Endo, Fuminori Yamasaki, Takashi Maeno, and Hiroaki Kitano. Co-evolving morphology and walking pattern of biped humanoid robot. In *International Conference on Robotics and Automation Washinton, DC May 2002*, 2002.

[13] J C Bongard and Chandana Paul. Making evolution an offer it can't refuse: Morphology and the extradimensional bypass. In *Sixth European Conference on Artificial Life , Prague, CZ*, 2003.

[14] Kalyanmoy Deb. *Multi-Objective Optimization using Evolutionary Algorithms.* Wiley, 2001.

[15] Chandana Paul. Bileteral decoupling in the neural control of biped locomotion. In *International Symposium on Adaptive Motion of Animals and Machines , Kyoto, Japan*, 2003.

[16] Chandana Paul. Sensorymotor control of biped locomotion based on contact information. In *International Symposium on Intelligent Signal Processing and Robotics to be held in Allahbad, India*, 2004.

[17] Fuminori Yamasaki, Tatsuya Matsui, Takahiro Miyashita, and Hiroaki Kitano. Pino the humanoid: A basic architecture. In *RoboCup*, pages 269–279, 2000.

[18] Peter Nordin and Mats Nordahl. Elvis: An evolutionary architecture for a humanoid robot. In *Proceeding of Symposium on Artificial Intelligence (CIMAF99)*, Havanna, Cuba, 1999.

[19] Fumio Kanehiro, Kenji Kaneko, Kiyoshi Fujiwara, Kensuke Harada, Shuuji Kajita, Kazuhito Yokoi, Hirohisa Hirukawa, Kazuhiko Akachi, and Takakatsu Isozumi. The first humanoid robot that has the same size as a human that can lie down and get up. In *Proceedings of the IEEE International Conference on Robotics and Automation, Taipei, Taiwan, 1633–1639*, 2003.

[20] Ruixiang Zhang and Prahlad Vadakkepat. An evolutionary algorithm for trajectory based gait generation of biped robot. Department of Electrical and Computer Engineering, National University of Singapore, 2003.

[21] http://world.honda.com/asimo/.

[22] Tim Pike. Gait generation for a humanoid robot, 2003.

[23] J. Teo and H.A. Abbass. Trading-off mind complexity and locomotion in a physically simulated quadruped. In L. Wang, K. Tan, T. Furuhashi, J. Kim, and X. Yao, editors, *Proceedings of the 4th Asia-Pacific Conference on Simulated Evolution And Learning (SEAL'02)*, volume 2, pages 776–780, 2002.